UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/660,353	09/11/2003	P. Anders I. Bertelrud	2095.001100/P3126	5128

62293 7590 01/28/2011

WILLIAMS, MORGAN & AMERSON, P.C.
10333 RICHMOND AVE.
SUITE 1100
HOUSTON, TX 77042

EXAMINER	
RUTTEN, JAMES D	
ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
01/28/2011	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/660,353	BERTELRUD ET AL.	
	Examiner	Art Unit	
	James D. Rutten	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *17 November 2010*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-4,6-13,15-21 and 23-37* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-4, 6-13, 15-21, and 23-37* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. The reply filed 11/17/2010 has been fully considered. Claims 1-4, 6-13, 15-21, and 23-37 are pending and have been examined.

Response to Arguments

2. Applicant's arguments on pages 8-16 filed 11/17/2010 have been fully considered but they are not persuasive.

Claim 9

On pages 8-9 filed 11/17/2010, Applicant argues that prior art McKeeman, col. 5, lines 21-23 teaches that code which is not changed/modified is not recompiled in contrast to claim 9 which calls for initiating compiling based on determining that the file was modified. Applicant essentially argues that not compiling unmodified code is not the same as initiating compilation of modified files. However, this interpretation neglects the surrounding passage which describes the compilation process in terms of initiating compilation but not upon files that have not changed or are not dependent upon files that have changed. In this case, there are a limited number of file states: changed, unchanged, and dependent-upon-changed. As such, by disclosing compilation, but not for files that are unchanged or not dependent-upon-changed, McKeeman is disclosing that compilation is initiated for files that have changed or are dependent-upon-changed. This disclosure anticipates the language of the claims.

On page 9, Applicant argues that McKeeman teaches away by teaching "compile time efficiency." This argument is not clearly presented since it is not clear why such a teaching would somehow discourage compilation of changed files. Applicant almost seems to argue that

McKeeman teaches away from recompilation of unchanged code. However, this is unlikely since no such claim limitations are present in claim 9. No further explanation is provided.

Claim 1

On pages 9-11 filed 11/17/2010, Applicant maintains the previous argument that the background compiling of prior art of record Robinson only performs parsing, and does not provide actual compilation. In particular, Applicant points to Robinson page 18 which mentions "the compiler parses." While it is agreed that Robinson clearly discloses parsing ability in the compiler, Applicant has not shown that Robinson's compiler is only a parser. Robinson clearly discloses a "Background Compiler" (see at least page 16). The fact that Robinson discloses a compiler with parsing ability does not nullify the disclosure of the background compiler which provides notice of "compilation errors." Applicant fails to persuasively explain why Robinson would use the term "compiler" in place of "parser." Furthermore, prior art of record "Advanced Basics; Scaling Up: The Very Busy Background Compiler," by Matthew Gertz, provides additional inherent details regarding Robinson's background compiler, and clearly describes traditional compilation through such topics as debugging, building executables and "drop it to disk." The Gertz document also discloses a "Compiled" state which occurs after the code is parsed. Parsing alone does not enable these technologies. In response to the above, Applicant argues that Gertz does not compile until code is "committed" by the user, and has equated such committing to a request to compile. However, the term "commit" is used in version control systems as a way of saving a changed file atomically from one state to the next (e.g. see http://en.wikipedia.org/wiki/Revision_control). There is no implicit suggestion of a compilation

request in the use of the term. Applicant has not persuasively shown that the term "commit" can be equated with a request to compile. Furthermore, the term "background compiler" itself suggests that compilation occurs in the background, i.e. without user intervention. Robinson's background compiler is capable of building an application without a request to compile, and is clearly more than a mere parser.

On page 12, Applicant essentially argues that McKeeman provides compilation in response to a user request, but not in response to determining that the file has been modified. While the cited portion of McKeeman (col. 5 lines 21-23 and surrounding text) does discuss a user request, it also explains that compilation occurs but not for modules that have not changed. As discussed above, this provides implicit disclosure of compiling not simply in response to a user request, but also in response to a determination that the file has changed. Furthermore, Robinson is relied upon to teach compilation in advance of a request from a user (see Robinson, page 16, e.g. "compiler that continually works in the background"). As noted above, Applicant's argument that Robinson does not teach compilation prior to a user's compilation request is not persuasive.

Near the bottom of page 12, Applicant points out that the claim recites "indicating a status of the compilation of the file in response to detecting the user request." This suggestion seems to be a continuation of previously submitted arguments that McKeeman fails to disclose "indicating a status of the compilation of the file in response to detecting the user request." The 8/17/2010 rejection cites McKeeman col. 5, lines 23-34 for disclosure of notification of compilation status. Compilation can occur in response to a user request (see McKeeman col. 5, lines 15-17). In this respect notification occurs as a result of, or in response to the initial

compilation request. The rejection also relies upon Robinson to teach compilation prior to a user request. As such, the combination of references teaches a McKeeman's status indication of Robinson's compilation.

In response to applicant's argument that the examiner's conclusion of obviousness is based upon improper hindsight reasoning (see pages 12-13 filed 11/17/2010), it must be recognized that any judgment on obviousness is in a sense necessarily a reconstruction based upon hindsight reasoning. But so long as it takes into account only knowledge which was within the level of ordinary skill at the time the claimed invention was made, and does not include knowledge gleaned only from the applicant's disclosure, such a reconstruction is proper. See In re McLaughlin, 443 F.2d 1392, 170 USPQ 209 (CCPA 1971). In response to applicant's argument that there is no teaching, suggestion, or motivation to combine the references, the examiner recognizes that obviousness may be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See In re Fine, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988), In re Jones, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992), and KSR International Co. v. Teleflex, Inc., 550 U.S. 398, 82 USPQ2d 1385 (2007). In this case, Robinson provides motivation to combine by teaching that background compilation provides a timely indication of errors (See Robinson page 16). Applicant essentially suggests that the teaching of Robinson fails to provide adequate motivation to combine. However, Applicant has not clearly explained why Robinson's discussion of a timely indication of errors would not

provide a motivation to combine. Robinson states "Visual Basic .NET has a compiler that continually works in the background. Compilation errors are flagged in code in real time with blue squiggle underlines." (See Robinson page 16). Such a description clearly suggests a time saving tool that one of ordinary skill in the art would instantly recognize. Furthermore, the section heading for discussion of Robinson's compiler is titled "Better Development Environment." One of ordinary skill in the art would be motivated to combine in order to provide such a better development environment.

Claim 2

On page 14, Applicant argues with respect to claim 2, that McKeeman fails to further disclose limitations found in claim 1. However, the rejection is provided in the context of the rejection of claim 1 which cites Robinson as teaching compiling in advance of a user request. The rejection of claim 2 includes the rejection of claim 1 which is based on a combination of McKeeman and Robinson. Applicant's argument is not persuasive for at least the reasons provided above with respect to claim 1.

Claim 3

On pages 14-15, Applicant argues with respect to new limitations found in claim 3, that neither McKeeman nor Robinson provide for "compiling the file to completion." However, McKeeman discloses complete compilation, at least as shown in Fig. 1. Applicant's arguments regarding Robinson are not persuasive, but are nonetheless moot in view of the disclosure of McKeeman. It is noted that the rejection is based upon the combined teachings of both

references, such that Robinson's background compilation provides at least a basis for McKeeman's complete compilation. As noted above, Applicant's argument that Robinson only teaches parsing is not persuasive.

Claims 30, 33, et al.

All further arguments are based upon prior arguments which are addressed above, and are not persuasive for the same reasons.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

4. Claims 9-13 and 15 are rejected under 35 U.S.C. 102(b) as being anticipated by U.S. Patent No. 5,193,191 (McKeeman et al.).

As per claim 9, McKeeman et al. discloses an article comprising one or more machine-readable storage media containing instructions (see, e.g., col. 8, lines 6-39) that when executed enable a processor to:

initiate compiling of a file including one or more code segments (see, e.g., col. 11, lines 44-61 (recompilation uses previously compiled code);

detect a user request to compile the file (see, e.g., Id. (recompilation is started)); and

provide a result associated with the compiling in response to detecting the user request (see, e.g., col. 5, lines 23-34 (errors are detected and reported));

wherein the instructions when executed enable the processor to initiate compiling of the file based on determining that the file was modified (see, e.g., col. 5, lines 21-23).

As per claim 10, McKeeman et al. further discloses the instructions when executed enable the processor to display a message to a user indicating that one or more errors were detected during the compiling (see, e.g., col. 5, lines 23-34 (errors are detected and reported)).

As per claim 11, McKeeman et al. further discloses the instructions when executed enable the processor to indicate to a user that the compiling was successful (see, e.g., col. 5, lines 23-34 (errors are detected and reported; if no errors are detected, then object code is produced as input to the linker)).

As per claim 12, McKeeman et al. further discloses the instructions when executed enable the processor to generate a file containing object code based on compiling the file and to store the object code file in a temporary location (see, e.g., col. 5, lines 30-34).

As per claim 13, McKeeman et al. further discloses the instructions when executed enable the processor to move the object code file from the temporary location into a product location based on determining that the compiling of the file was successful and in response to detecting the user request (see, e.g., col. 5, lines 23-34 (errors are detected and reported; if no errors are detected, then object code is produced as input to the linker)).

As per claim 15, McKeeman et al. further discloses the instructions that when executed enable the processor to initiate compiling of the file based on determining that the file was

modified comprise instructions that when executed enable the processor to indicate in a work queue that the file has been modified and to initiate compiling of the file in response to detecting the indication (see, e.g., col. 11, lines 44-61).

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 1-4, 6-8, 16-21, 23-29, 31-32, and 35-37 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,193,191 (McKeeman et al.) in view of "Upgrading Microsoft Visual Basic 6.0 to Microsoft Visual Basic .NET" by (Robinson et al.).

As per claim 1, McKeeman et al. discloses:

initiating compilation of a file in a processor-based system … (see, e.g., col. 11, lines 44-61 (recompilation uses previously compiled code));

McKeeman et al. does not expressly disclose: in advance of a request from a user to compile the file. However, Robinson et al. teaches background compilation in advance of a compilation request (see page 16 "Visual Basic .NET has a compiler that continually works in the background.") It would have been obvious to one of ordinary skill in the art at the time the invention was made to use the compilation of McKeeman et al. with the background compiler of Robinson et al. in order to provide a timely indication of errors as suggested by Robinson et al..

detecting the user request to compile the file (see, e.g., Id. (recompilation is started)); and

indicating a status of the compilation of the file in response to detecting the user request (see, e.g., col. 5, lines 23-34 (errors are detected and reported));

wherein initiating compilation of the file comprises compiling the file in response to determining that the file has been modified (see, e.g., col. 5, lines 21-23).

As per claim 2, McKeeman et al. further discloses initiating compilation of the file comprising compiling the file including one or more code segments to produce an object code file (see, e.g., col. 5, lines 30-34).

As per claim 3, McKeeman et al. further discloses compiling the file comprising compiling one or more code segments in the file to produce an object code file, and further comprising linking the object code file to produce an executable file (see, e.g., col. 5, lines 37-46), and wherein initiating compilation of a file in a processor-based system in advance of a request from a user to compile the file further comprises compiling the file to completion (see McKeeman Fig. 1 which depicts completion not only of object files, but also of an executable image; also see associated text in column 5).

As per claim 4, McKeeman et al. further discloses indicating the status of the compilation of the file comprising at least one of indicating that the compilation was successful and indicating that the compilation was unsuccessful (see, e.g., col. 5, lines 23-34).

As per claim 6, McKeeman et al. further discloses determining that the file has been modified comprising determining that the modified file has been saved to a storage unit (see, e.g., col. 5, lines 18-23).

As per claim 7, McKeeman et al. further discloses the file including one or more code segments, wherein initiating compilation of the file in response to determining that the file has been modified comprises:

identifying the modified file in a work queue (see, e.g., col. 11, lines 44-61); and

initiating the compilation of the file based on the modified file being identified in the work queue (see, e.g., col. 11, lines 44-61).

As per claim 8, McKeeman et al. further discloses indicating the status of the compilation of the file comprising generating one or more files associated with the compilation of the file, storing the one or more generated files in a temporary location, and transferring the one or more files from the temporary location to a different location in response to detecting the user request (see, e.g., col. 5, lines 30-34).

As per claim 16, McKeeman et al. discloses an apparatus (see, e.g., col. 8, lines 6-39), comprising:

means for initiating compilation of a file in a processor-based system …(see, e.g., col. 11, lines 44-61 (recompilation uses previously compiled code));

McKeeman does not expressly disclose: in advance of a request from a user. However, Robinson et al. teaches background compilation in advance of a compilation request (see page 16 "Visual Basic .NET has a compiler that continually works in the background.") It would have been obvious to one of ordinary skill in the art at the time the invention was made to use the compilation of McKeeman et al. with the background compiler of Robinson et al. in order to provide a timely indication of errors as suggested by Robinson et al..

means for detecting the user request to compile the file (see, e.g., Id. (recompilation is started)); and

means for indicating a status of the compilation of the file in response to detecting the user request (see, e.g., col. 5, lines 23-34 (errors are detected and reported));

wherein the means for initiating compilation initiate compiling the file based on determining that the file was modified (see, e.g., col. 5, lines 18-23; col. 13, lines 42-54).

McKeeman et al. further discloses a machine-readable storage media containing instructions for implementing the recited functionality (see, e.g., col. 8, lines 6-39).

As per claim 17, McKeeman et al. discloses an apparatus (see, e.g., col. 8, lines 6-39), comprising:

a storage unit having a file stored therein (see, e.g., col. 8, lines 6-39); and

a control unit communicatively coupled to the storage unit (see, e.g., col. 8, lines 6-39), the control unit adapted to:

initiate compilation of the file (see, e.g., col. 11, lines 44-61 (recompilation uses previously compiled code));

McKeeman does not expressly disclose: in advance of a request from a user to compile the file. However, Robinson et al. teaches background compilation in advance of a compilation request (see page 16 "Visual Basic .NET has a compiler that continually works in the background.") It would have been obvious to one of ordinary skill in the art at the time the invention was made to use the compilation of McKeeman et al. with the

background compiler of Robinson et al. in order to provide a timely indication of errors as suggested by Robinson et al..

detect the user request to compile the file (see, e.g., Id. (recompilation is started)); and

indicate a status of the compilation of the file in response to detecting the user request (see, e.g., col. 5, lines 23-34 (errors are detected and reported));

wherein the control unit is adapted to compile the file in response to determining that the file has been modified (see, e.g., col. 5, lines 18-23; col. 13, lines 42-54).

As per claim 18, McKeeman et al. discloses the control unit is adapted to compile a file including one or more code segments to produce an object code file (see, e.g., col. 5, lines 30-34).

As per claim 19, McKeeman et al. discloses the control unit is adapted to link the object code file to produce an executable file (see, e.g., col. 5, lines 37-46).

As per claim 20, McKeeman et al. discloses the control unit is adapted to store the executable file in a temporary location and to transfer the executable file from the temporary location to a different location based on detecting the user request (see, e.g., col. 5, lines 30-34).

As per claim 21, McKeeman et al. discloses the control unit is adapted to at least one of indicate that the compilation was successful and indicate that the compilation was unsuccessful (see, e.g., col. 5, lines 23-34).

As per claim 23, McKeeman et al. discloses the control adaptation to compile the file in response to determining that the file has been modified comprises:

an adaptation to identify the modified file in a work queue (see, e.g., col. 11, lines 44-61); and

an adaptation to initiate the processing of the file based on the modified file being identified in the work queue (see, e.g., col. 11, lines 44-61).

As per claim 24, McKeeman et al. discloses:

identifying one or more source files that have been modified in a processor-based system (see, e.g., col. 5, lines 18-23; col. 13, lines 42-54);

initiating processing of at least a portion of the modified source files (see, e.g., col. 11, lines 44-61);

McKeeman does not expressly disclose: before receiving a request to process the modified files. However, Robinson et al. teaches background compilation in advance of a compilation request (see page 16 "Visual Basic .NET has a compiler that continually works in the background.") It would have been obvious to one of ordinary skill in the art at the time the invention was made to use the compilation of McKeeman et al. with the background compiler of Robinson et al. in order to provide a timely indication of errors as suggested by Robinson et al..

receiving the request to process at least one of the modified files (see, e.g., col. 11, lines 44-61 (recompilation uses previously compiled code)); and

providing a status associated with the processing of the file in response to receiving the request (see, e.g., col. 5, lines 23-34 (errors are detected and reported)).

As per claim 25, McKeeman et al. further discloses the processor-based system is adapted to execute an integrated development environment module (see, e.g., col. 3, lines 53-56), wherein identifying the one or more files comprises the integrated development environment module placing the one or more of the source files that have been modified in a queue (see, e.g., col. 11, lines 44-61).

As per claim 26, McKeeman et al. further discloses placing the one or more of the source files in the queue comprises placing at least one source file in the queue in response to a user saving the source file to a storage unit (see, e.g., col. 5, lines 18-23).

As per claim 27, McKeeman et al. further discloses placing the one or more of the source files in the queue comprises placing at least a portion of one source file in the queue in response to a user saving the source file to a storage unit using an editor and then exiting from the editor (see, e.g., col. 17, lines 11-36).

As per claim 28, McKeeman et al. further discloses placing the one or more of the source files in the queue comprises placing at least one source file in the queue in response to determining that a user desires to compile at least a portion of the source file as the source file is being edited (see, e.g., col. 5, lines 15-17; col. 11, lines 44-61).

As per claim 29, McKeeman et al. further discloses placing the one or more of the source files in the queue comprises placing at least one source file in the queue in response to determining that the source file includes at least one marker identifying a section of the source file that should be compiled, and wherein initiating processing of at least the portion of the one or more modified files comprises compiling the identified section of the source file (see, e.g., col. 11, lines 44-61).

As per claim 31, McKeeman et al. further discloses initiating the processing comprises initiating a build process to produce a software application and wherein receiving the request comprises receiving the request to building the software application (see, e.g., col. 5, lines 15-17).

As per claim 32, McKeeman et al. further discloses initiating the build process comprises performing compiling the modified source files to produce object code files and linking the object code files to produce executable files (see, e.g., col. 5, lines 37-46); and wherein initiating compiling further comprises compiling the file to completion (see McKeeman Fig. 1 which depicts completion not only of object files, but also of an executable image; also see associated text in column 5).

As per claim 35, McKeeman et al. further discloses the object code files and the executable files are moved to a different storage location in response to detecting the request and in response to detecting no error or warning (see, e.g., col. 5, lines 23-34 (errors are detected and reported; if no errors are detected, then object code is produced as input to the linker)).

As per claim 36, McKeeman et al. further discloses identifying one or more source files comprises identifying the one or more source files based on a directed acyclic graph (see, e.g., col. 16, 54-63).

As per claim 37, McKeeman et al. further discloses the directed acyclic graph includes a list of dependent files, wherein identifying one or more source files comprises identifying at least one modified source file and another source file that is dependent on the modified source file using the directed acyclic graph (see, e.g., col. 16, 54-63).

7. Claim 30 is rejected under 35 U.S.C. 103(a) as being unpatentable over McKeeman et al. and Robinson et al. as applied above, and further in view of U.S. Pat. App. Pub. No. 2005/0108682 (Piehler et al.).

As per claim 30, McKeeman et al. discloses such a method (see the rejection of claims 24 and 25 under 35 U.S.C. § 102(b)) but fails to expressly disclose initiating the processing of the modified source files comprises causing a background thread to awaken in response to placing the one or more of the source files in the queue, where the background thread thereafter initiates processing of the source files. However, Piehler et al. teaches such use of a background thread to enqueue a task for itself to complete the rest of the compilation in a background thread as part of a response to a new file being added to a project or an existing file being modified (Piehler et al. at paragraphs [0170] through [0180]. Therefore, it would have been obvious to include such background thread use as per the teachings of Piehler et al. One would be motivated to do so to gain the advantage of providing immediate feedback to the user without any detectable visible delay in typing responsiveness while the compiler finishes processing the change (Poehler et al. at paragraph [0174]).

8. Claim 33 is rejected under 35 U.S.C. 103(a) as being unpatentable over McKeeman et al. and Robinson et al. as applied above, and further in view of U.S. Pat. No. 6,230,313 (Callahan, II et al.).

As per claim 33, McKeeman et al. further discloses placing the one or more of the source files in the queue comprising placing at least one source file in the queue in response to determining that the source file includes …[markers] identifying a section of the source file that

should be compiled, wherein the … marker defines the beginning of the portion of the source file and the … marker defines the end of the portion, and wherein initiating processing of at least the portion of the one or more modified files comprises compiling the identified section of the source file (see, e.g., col. 16, line 60, through col. 17, line 6 (the compiler of McKeeman can identify changed portions (having a defined beginning and end) within a source code file and recompile only those changed portions)). McKeeman does not expressly disclose: at least two markers. However, Callahan, II et al. teaches the use of markers at the beginning and end of a region of interest (see column 10 lines 10-25). It would have been obvious to one of ordinary skill in the art at the time the invention was made to use McKeeman's markers with Callahan's teaching of multiple markers in order to easily identify regions of interest as suggested by Callahan.

9. Claim 34 is rejected under 35 U.S.C. 103(a) as being unpatentable over McKeeman et al. and Robinson et al. as applied above, and further in view of prior art of record U.S. Patent Application Publication US 2005/0114771 by Piehler et al. ("Piehler").

As per claim 34, McKeeman et al. discloses indication of errors (see, e.g., col. 5, lines 23-34 (errors are detected and reported)). McKeeman does not expressly disclose: suppressing at least one of an error and warning that is detected while compiling the modified source files. However, Piehler teaches suppression of error indication. See at least paragraphs [0034]-[0036]. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use McKeeman's error indication with Piehler's error suppression in order to make code completion more robust as suggested by Piehler (see paragraph [0035]).

Conclusion

10. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Matthew Gertz, "Advanced Basics; Scaling Up: The Very Busy Background Compiler," Microsoft Corp., MSDN Magazine, June 2005, 4 pages, discusses in more detail the implementation of the VB.NET background compiler and the changes in its design from the VISUAL BASIC .NET 2002 development environment to the VISUAL BASIC .NET 2005 development environment.

11. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

12. Any inquiry concerning this communication or earlier communications from the examiner should be directed to James D. Rutten whose telephone number is (571)272-3703. The examiner can normally be reached on M-F 10:00-6:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571)272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/J. Derek Rutten/
Primary Examiner, Art Unit 2192